Czech University of Life Sciences

PRAGUE

Faculty of Economics and Management

Informatics

Department of Information Engineering

Diploma Thesis

Modeling And Simulation of Automated Teller Machine (ATM)

Author: Raphael Kwaku Botchway
Supervisor: Dr. Ing. Robert Pergl

DECLARATION

I declare that I have worked on my diploma thesis titled **"Modeling and Simulation of Automated Teller Machine (ATM)"** by myself and I have only used sources mentioned at the end of the thesis.

In Prague ....

Raphael Kwaku Botchway

ACKNOWLEDGEMENT

My heartfelt gratitude goes to my supervisor Dr. Ing. Robert Pergl,who provided me guidance and support throughout the project. I specially appreciate his willingness to help anytime. Special thanks to God Almighty for His gift of knowledge, wisdom and understanding that has brought me this far. My finally thanks goes to my kids Ben Caleb, Freda as well as my wife Mabel for their continuous moral support and encouragement.

Modeling And Simulation of Automated Teller Machine (ATM)

Modelování a simulace bankomatu

Modeling And Simulation Of Automated Teller Machine (ATM)

Summary

The introduction of the Automated Teller Machine (ATM) in banking has brought about enormous benefits to both customers and banks. Its management involves several processes and data that need to be described and modeled in an engineering approach. We provide a brief description of the ATM and its activities as a starting point. By using the Unified Modeling Language (UML), we then design a set of engineering models for the ATM system.

Using Petri Nets, we also design another set of models and simulations for some selected activities of the Automated Teller Machine used by banks to serve its customers. Activities modeled include Cash Withdrawal, Funds Transfer, Balance Enquiry and Mobile Top-Up (Credit).

Petri Nets provides a graphical and mathematical tool with a uniform environment for modeling, formal analysis and design of discrete event-driven systems. Its simulation functionality distinguishes it from other modeling languages like the Unified Modeling Language.

With the aid of HPSim software, we modeled and simulated all the Petri Net models considered. Finally, the Producer-Consumer problem is discussed and optimization possibilities through efficient resource allocation and management are formulated.

Keywords

UML, ATM, Petri Nets, modeling, simulation, optimization, HPSim

Modelování a simulace bankomat

Souhrn

Zavedení bankomatů (ATM) v bankovním sektoru přinesl obrovské výhody pro zákazníky a bankami. Provozování bankomatů představuje řadu procesů a dat, které je třeba mít popsáno a namodelováno s použitím inženýrských přístupů. Práce začíná stručným popisem ATM a jeho činnosti, poté je vytvořen soubor modelů v notaci Unified Modeling Language (UML).

Tato práce dále využívá aparátu Petriho sítí pro návrh souboru modelů a simulací pro některé vybrané činnosti běžných bankomatů. Modelovány byly výběr hotovosti, převodu finančních prostředků, zjištění zůstatku na účtu a služeb mobilních operátorů (dobíjení).

Aparát Petriho sítí představuje grafický a matematický nástroj pro modelování, formální analýzu, návrh a simulace diskrétních systémů řízených událostmi. Tyto možnosti jej odlišují od ostatních modelovací notací, jako je Unified Modeling Language.

S pomocí softwaru HPSim byla provedena simulace všech navržených Petriho sítí. V závěru práce je též diskutován problém producent-konzument v souvislosti s problematikou dostupných prostředků v bankomatu a je diskutována optimalizace prostřednictvím efektivní alokace zdrojů a řízení.

Klíčová slova

UML, ATM, Petriho sítě, modelování, simulace, optimalizace,HPSim

Contents

1. Introduction

A critical look at the introduction of the Automated Teller Machine (ATM) as part of the electronic revolution that has engulfed modern banking operations since 1967 has brought about enormous benefits to both customers and banks. To the banks, they have been relieved of constraints such as time and geographical location.

Aside providing customers with cash, they can also carry out certain banking operations such as deposits and mobile phone top up. The operational activities of an automated teller machine (ATM) can be modeled using the unified modeling language (UML).

For the purposes of this thesis, modeling the ATM activities/operations using UML methodologies such as class diagrams, use case and sequence diagrams will serve as a prelude to the major discussion on modeling ATM operations using Petri Nets. The crust of this thesis will focus on modeling and simulating the ATM activities using Petri Nets (PN).

Petri Nets was developed originally by Carl Adam Petri in 1962 as a graphical and mathematical modeling tool applicable to many systems. It is a promising tool for describing and studying information processing systems that are characterized as being concurrent, synchronous, distributed, parallel, nondeterministic, and/or stochastic.

As a graphical tool, Petri Nets can be used as a visual-communication aid similar to flow charts, block diagrams and networks. The graphical representation of Petri Net models consist of a network formed by places, transitions and arcs.

Mathematically, it is possible to set up state equations, algebraic equations and other mathematical models governing the behavior of systems. In addition, tokens are used in these nets to simulate the dynamic and concurrent activities of systems. The mode of operation of the ATM that is modeled by the Petri Net is simulated using HPSim software (Petri Net software).

1.1 Objectives

The major goals of this thesis include:

- Designing a set of models and simulations for the Automated Teller Machine and its servicing processes.
- Formulating conclusions about optimization possibilities through efficient resource allocation and management.

1.2 Methodology

The project is basically divided into two (2) parts namely i) literature review and ii) formulation of results. Invited tutorial-review papers and proceedings on Petri Nets from scientific conferences, internet resources and other literature sources would constitute my main source of information (literature review).

Most of the discussion under the initial part will be centered the history of the ATM, a description of how it operates, the benefits to the banking industry as well any likely disadvantages if any. The next step of the project will focus on the formulation of results which will consist of the following steps:

- Data collection: Contacting various banks with the view to gather information about the various activities of their respective ATM's.
- Data analysis: Analyse the data collected by narrowing down on the major uses of the ATM in the banking sector.
- Synthesis of data (model design): Develop an appropriate model(s) for the ATM with special emphasis on its business processes.
- Verification of the model: This ensures that the model(s) is developed in conformity with the model specification and is implemented correctly without errors.
- Design simulation for the models: Use Petri Net software (HPSim) to simulate the models that have been developed for all the ATM's business processes.
- Formulate conclusions: Make useful conclusions on the work performed so far and offer suggestions which can act as the basis for further research.

2. The Automated Teller Machine (ATM)

2.1 History

The history of ATM can be traced back to the 1960s, when the first ATM machine was invented by Scot John Shepherd-Barron and used by Barclays Bank in 1967. However, while Shepherd-Barron has the major claim to fame, there have been many other individuals who have also invented some version of the ATM.

The machine itself has evolved over the years, with the earlier versions restricted to only one or few banking functions. There has been much debate, however, on who invented the first early versions of Automated Teller Machine. But the history of ATM can be visibly traced back to the year of 1967.

In 1939, a rudimentary cash dispenser was invented by Luther George Simijian and established by the City Bank of New York. However, the machine did not work much and had to be removed within six months of putting up the machine. The early versions of the ATM were restricted to cash withdrawal only.

In the 1967 model, patented by Shepherd-Barron, the plastic cards did not exist and instead, vouchers with a strip of radioactive substance were used for withdrawing cash. Consequently, the vouchers were matched with a particular personal pin code used by the bank to identify the customer.

The ATM was inaugurated by renowned British actor, Reg Varney. The personal identification number was initially a six numbered password, and was later changed to a four numbered password. However, this automated teller machine was very different from the

modern day teller machines, which is based on an electronic system between the different branches of the bank. Thus, the history of ATM has seen many changes over the span of 25 years since 1939.

Pictures of old and new ATM

Figure 2.1 Old ATM

Figure 2.2 Modern ATM

2.2 Parts of the Automated Teller Machine

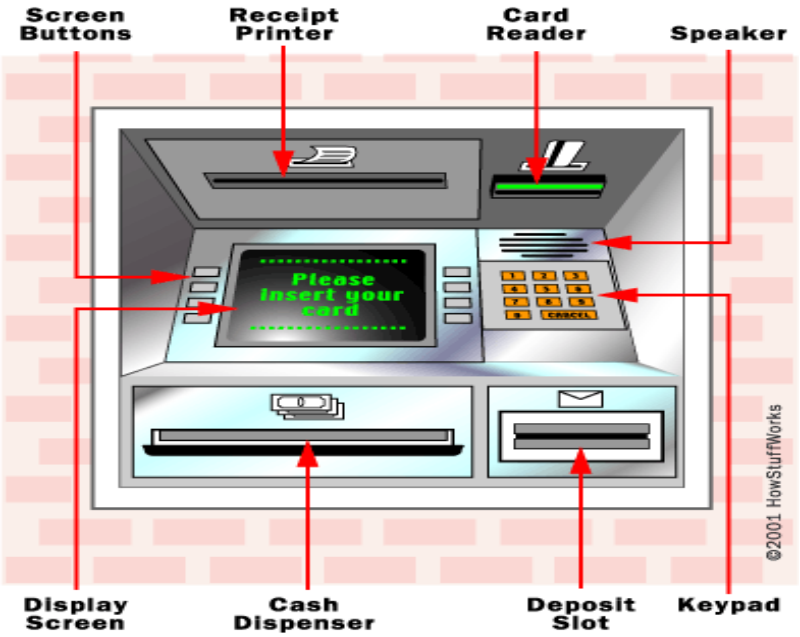

Fig 2.3 Parts of ATM

An ATM has two input devices namely:

- **Card reader** - The card reader captures the account information stored on the magnetic stripe on the back of an ATM/debit or credit card. The host processor uses this information to route the transaction to the cardholder's bank.
- **Keypad** - The **keypad** lets the cardholder tell the bank what kind of transaction is required (cash withdrawal, balance inquiry, etc.) and for what amount. Also, the bank requires the cardholder's personal identification number (**PIN**) for verification. Federal law requires that the PIN block be sent to the host processor in encrypted form.

Its output devices include:

- **Speaker** - The speaker provides the cardholder with auditory feedback when a key is pressed.
- **Display screen** - The display screen prompts the cardholder through each step of the transaction process. Leased-line machines commonly use a monochrome or color CRT (cathode ray tube) display. Dial-up machines commonly use a monochrome or color LCD.
- **Receipt printer** - The receipt printer provides the cardholder with a paper receipt of the transaction.
- **Cash dispenser** - The heart of an ATM is the safe and cash-dispensing mechanism. The entire bottom portion of most small ATMs is a safe that contains the cash.

ATM uses include:

- Withdrawing cash
- Depositing cash
- Depositing cheques
- Checking account balance (balance Enquiry)
- Account mini statement
- Mobile recharge/top up etc…

3. Petri Nets

3.1 Introduction

Petri Nets was originally developed by Carl Adam Petri [Pet62], and was the subject of his dissertation in 1962 presented at the faculty of Mathematics and Physics at the Technical University of Darmstadt. He prepared his dissertation when he worked as a scientist at the University of Bonn.

Petri Nets are a graphical and mathematical modeling tool applicable to a variety of systems and in many areas. They are a promising tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic.

As a graphical tool, Petri Nets can be used as a visual-communication aid similar to flow charts, block diagrams and Networks. Additionally, tokens are used in these Nets to simulate the dynamic and concurrent activities of systems. Mathematically, Petri Nets can be used as a tool to set up state equations, algebraic equations and other mathematical models to govern the behavior of systems.

3.2 Classification Of Petri Nets

For the past 40 years, Petri Nets have seen some major developments. There are several types of Petri Nets with each having some more features than the earlier ones. These Petri Nets are often divided into two groups namely:

- Classical Petri Nets
- High Level Petri Nets

High Level Petri Nets

Classical Petri Net does not allow for the modeling of data and time. To solve these problems, many extensions to Petri Nets have been developed. Three (3) well-known extensions of the basic Petri Net model are the:

- extension with color to model data
- extension with time, and
- extension with hierarchy

Extension with Color (Colored Petri Nets)

Tokens often represent objects (e.g. resources, goods, humans) in the modeled system. Hence, we often want to represent attributes of these objects. If an insurance claim is modeled by a token in the Petri Net, we may want to represent attributes such as the name of the claimant, identification number, date, and amount. Since these attributes are not easily

represented by a token in a classical Petri Net, we extend the Petri Net model with colored or typed tokens. In a colored Petri Net each token has a value often referred to as `color'.

Transitions determine the values of the produced tokens on the basis of the values of the consumed tokens, i.e. a transition describes the relation between the values of the input tokens' and the values of the `output tokens'. It is also possible to specify `preconditions' which take the colors of tokens to be consumed into account.

Extension with Time (Timed Petri Nets)

For real systems it is often important to describe the temporal behavior of the system, i.e., we need to model duration and delays. Since the classical Petri Net is not capable of handling quantitative time, a timing concept is added. There are many ways to introduce time into the Petri Net. Time can be associated with tokens, places, and/or transitions.

In applying Petri Nets formally to the quantitative analysis of the performance and reliability of system with respect to time, a class of Petri Nets called Time Petri Nets is used. The time variables associated to the Petri Nets can be either deterministic variables (deterministic Petri Nets), or random variables (Stochastic Petri Nets).

Extension with Hierarchy

Although timed colored Petri Nets allow for a succinct description of many business processes, precise specifications for real systems have a tendency to become large and complex. This is the reason we provide a hierarchy construct, called sub Net.

A sub Net is an aggregate of a number of places, transitions, and subsystems. Such a construct can be used to structure large processes. At one level we want to give a simple description of the process (without having to consider all the details). At another level we want to specify a more detailed behavior. The extension with hierarchy allows for such an approach.

3.3 Petri Net Concepts

A Petri Net is a particular kind of directed graph, together with an initial state called initial Marking M_0.In simple terms it's a collection of directed arcs connecting places to transitions. The graph *N* of a Petri Net is directed, weighted and a bipartite graph consisting of two (2) kinds of nodes namely *places* and *transitions*.

By rule arcs connect places to transitions and vice versa. Graphically, places and transitions are denoted by circles and boxes or bars respectively. Arcs are labeled with their weights (positive integers) and have a capacity of one (1) by default. Places have infinite capacity by default and transitions have no capacity and cannot store tokens.

Formal Definition of a Petri Net

A Petri Net is a 5-Tuple, PN = (P, T, F, W, M_o) where

- P is a finite set of Places
- T is a finite set of Transitions
- The places P and Transitions T are disjoint ($P \cap T = \emptyset$)
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs(flow relation)
- W: $F \rightarrow \{1, 2, 3\ldots\}$ is a weight function
- M_o: $P \rightarrow \{0, 1, 2, 3\ldots\ldots\}$ is the initial marking representing initial distribution of tokens.

A Petri Net structure $N = (P, T, F, W)$ without any specific initial marking is denoted by N whereas a Petri Net with a given initial marking is represented as (N, M_o).

If $\langle p, t\rangle \in F$ for a transition t and a place p, then p is an input place of t. If $\langle t, p\rangle \in F$ for a transition t and a place p, then p is an output place of t. Additional notations used for pre-set and post-set are as follows:

- $\bullet t = \{ p | (p, t) \in F \}$ = the set of input places of t
- $t\bullet = \{ p | (t, p) \in F \}$ = the set of output places of t
- $\bullet p = \{ p | (t, p) \in F \}$ = the set of input transitions of p
- $p\bullet = \{ p | (p, t) \in F \}$ = the set of output transitions of p

Transition Enabling and Firing

A state or marking in a Petri Net is changed according to the following transition (firing) rule:

- A transition t is said to be enabled if each input place p of t is marked with at least w(p, t) tokens, where w(p, t) is the weight of the arc from p to t.
- An enabled transition may or may not fire (depending on whether or not the event actually takes place).
- A firing of an enabled transition t removes w(p, t) tokens from each input place p of t, and adds w(p, t) tokens to each output place p of t, where w(t, p) is the weight of the arc from t to p.

When a transition *t* is fired, the tokens in the input places are moved to output places, according to their arc weights and place capabilities.

Input Places	**Transition**	**Output Place**
Preconditions	Event	Post Conditions
Input Data	Computation Step	Output Data
Input Signals	Signal processor	Output Signals
Resources Needed	Task/Job	Resources released
Conditions	Logic Clause	Conclusions
Buffers	Processor	Buffer

Table 3.1 Some interpretations of transitions and places

Consider the Petri Net drawn below.

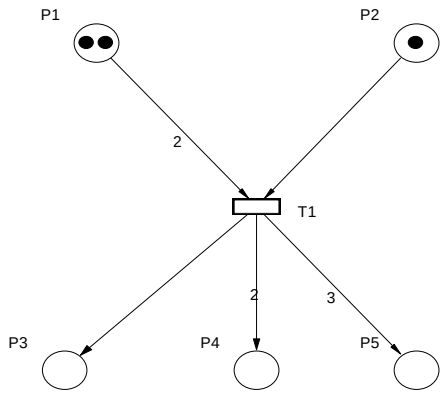

Figure 3.1

The Petri Net PN = (P, T, F, W, $M_{0)}$ for Figure 3.1 is defined as follows:

P = {P1, P2, P3, P4, P5}

T = {T1}

F = { <P1,T1>,<P2,T1>,<T1,P3>,<T1,P4>,<T1,P5> }

W = {<P1, T1 > ⟶2, <P2, T1 > ⟶1, <T1, P3 > ⟶1, <T1,P4 > ⟶2,<T1,P5 > ⟶3}

M_0= {P1 ⟶2, P2 ⟶1}

The Petri Net or Place/Transition Net (PN) when enabled and fired will produce the Petri Net marking in

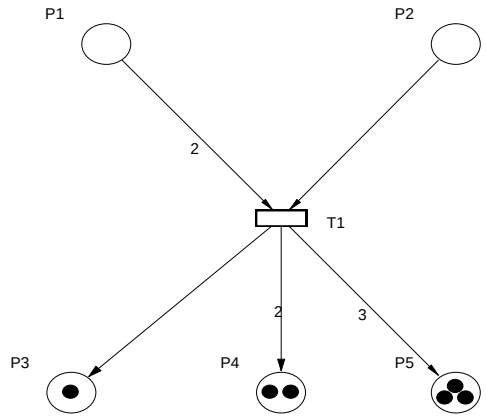

Figure 3.2

By rule a transition is enabled when the number of tokens in each of its input place is at least equal to the arc weight going from the place to the transition. It must be noted an

enabled transition can fire at anytime. When the transition fires, it takes the tokens from their input places and distributes the tokens to output places according to their arc weights.

Using the rule of transition enabling, it is sometimes assumed that each place in the Petri Net can accommodate an infinite number of tokens. A Petri Nets of this nature is called an *Infinite Capacity Petri Net*. When modeling many physical systems, we consider an upper limit to the number of tokens that each place can hold. This kind of Net is called a *Finite Capacity Net*.

Each place *p* of a finite capacity Net has an associated capacity *K (p)* that indicates the maximum number of tokens that *p* can hold. Similarly for a transition *t* to be enabled in a *Finite Capacity Net* there is an additional condition that the number of tokens in each output place *p* of *t* cannot exceed its capacity *K (p)* after it's fired. This rule with a capacity constraint is called the *Strict Transition Rule* .A special type of *inhibitor arc* is used to reverse the logic of an input place.

The *inhibitor arc* connects a place to a transition and is represented by a dashed line terminating with a small circle instead of an arrowhead at the transition. The class of Petri Nets with inhibitor arcs is referred to as *Extended Petri Nets*.

3.4 Petri Net Structures

Let us consider Figure 3.3.

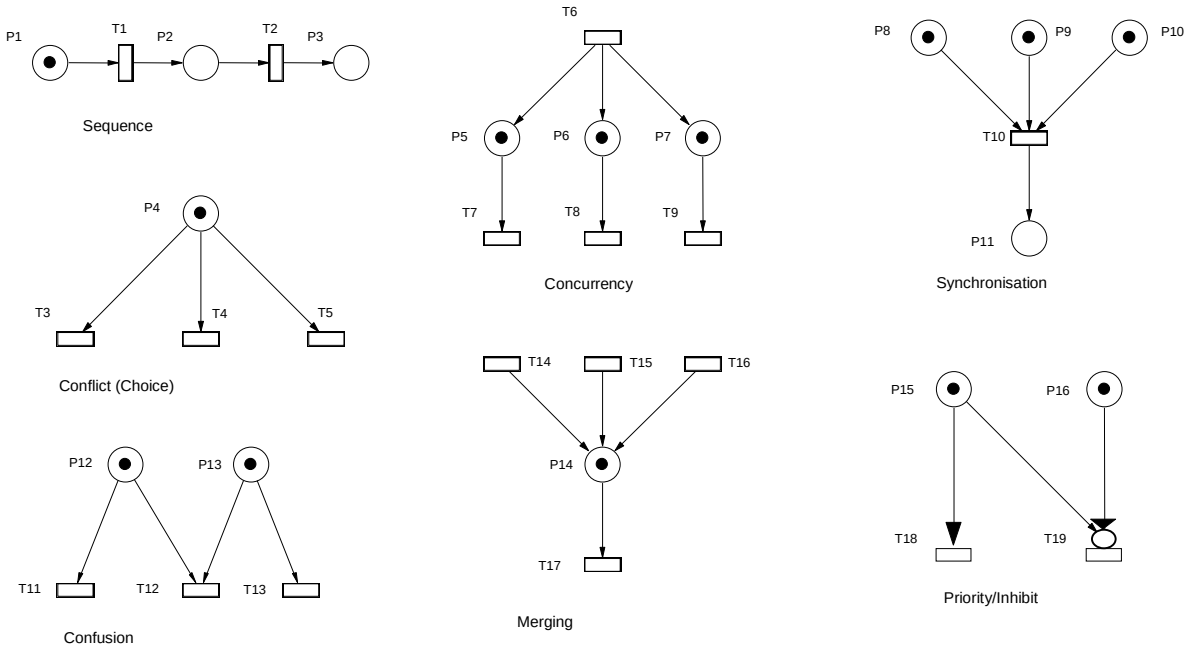

Figure 3.3

Sequence

- Transition T2 can only fire only after the firing of T1.

Conflict

- Token P4 can enable either transition T3, T4 or T5.Any transition that fires first automatically disables the remaining two.

Concurrency

- Tokens P5, P6, P7 can enable transitions T7, T8 and T9 concurrently.

Synchronization

- When processes leading into P8, P9 and P10 are finished, they are synchronized by starting P11.

Confusion

- Is the combination of conflict and concurrency.P12 enables both T11 and T12. However if T11 fires, T12 become disabled.

Merging

- Basically merges the three parallel processes.

Priority/Inhibit

- Uses the inhibit arc to control T19.As long as P16 has a token P19 cannot fire.

3.5 Behavioral Properties

Reachability

The firing of an enabled transition will change the token distribution in a Net according to the transition rule. A marking M_n is said to be reachable from a marking M_0 if there exist a sequence of firings

$\sigma = M_0\ t_1\ M_1\ t_2\ M_3\ t_2 \ldots\ t_n\ M_n$ that transforms M_0 to M_n. In this case, M_n is reachable from M_0. The set of all possible markings reachable from M_0 in a Net (N, M_0) is denoted by

$R(N,M_0)$ or $R(M_0)$ whereas $L(N, M_0)$ denotes the set of all possible firing sequences from M_0. The reachability problem is decidable.

Boundedness

A Petri Net (N, M_0) is said to be k-bounded or simply bounded if the number of tokens in each place does not exceed a finite number k for any marking reachable from M_0, i.e. $M(p) \leq k$ for every place p and every marking $M \in R(M_0)$.A Petri Net (N, M_0) is said to be safe bounded if it is 1-bounded.

By verifying that a Net is safe or bounded, it is guaranteed that there will be no overflows in the buffers or registers, no matter what firing sequence is taken. Places in Petri Nets are often used to represent buffers and registers used to store intermediate data.

Liveness

A Petri Net (N, M_0) is said to be live if no matter what marking has been reached from M_0, it is possible to ultimately fire any transition of the Net by progressing through some further firing sequence. This means that a live Petri Net guarantees deadlock-free operation. There are different levels of liveness.

A transition *t* in a Petri Net (N, M_0) is said to be

- Dead (live0-live) if *t* can never be fired in any firing sequence in $L(M_0)$.
- L1-live(potentially firable) if *t* can be fired at least once in some firing sequence in $L(M_0)$
- Live2-live if, given any positive integer *k*, *t* can be fired at least k times in some firing sequence in $L(M_0)$.
- L3-live if *t* appears infinitely, often in some firing sequence in $L(M_0)$.
- L4-live or live if *t* is L1-live for every marking in $R(M_0)$.

Coverability

A marking M in a Petri Net (N, M_0) is said to be coverable if there exists a marking M' in $R(M_0)$ such that $M'(p) \geq M(p)$ for each p in the Net. Coverability is closely related L1-liveness (potential firable). Let M be the minimum marking needed to enable a transition *t*. Then *t* is dead (L0-live) if and only if M is not coverable. However, *t* is L1-live if and only of M is coverable.

3.6 Analysis Methods

Methods of analysis for Petri Nets may be classified into the three groups namely:

- the coverability (reachability) tree method
- the matrix-equation approach
- reduction or decomposition approach

The first method deals with enumeration of all reachable markings or their coverable markings. It should be able to apply to all classes of nets, but is restricted to “small” nets due to the complexity of the state-space explosion.

On the other hand, matrix equations and reduction techniques are powerful but in many cases are they are applicable only to special subclasses of Petri nets or special situations.
For the purposes of this diploma thesis I would like to briefly describe only the coverability or reachability tree and its graph.

Coverability Tree and Graph

For a given Petri Net (N, M_0) with initial marking M_0, it is possible to obtain *new* markings as the number of the enabled transitions. From each new marking, we can again obtain reach more markings. This process produces a tree representation of the markings. Nodes represent markings generated from M_0 (the root) and its successors, and each a transition firing, which transforms one marking to another. This tree representation will grow infinitely large if the Net is unbounded.

A special symbol ω known as infinity is used to keep the tree finite. The symbol ω is defined such that for each positive integer n,

$\omega > n$, $\omega \pm n = \omega$ and $\omega \geq \omega$.

For a bounded Petri Net, the coverability tree is called the reachability tree since it contains all possible reachable markings. The coverability tree for a Petri Net (N, M_0) is constructed using an algorithm.

Algorithm

Step 1

Label the initial marking M_0 as the root and tag it “new”.

Step 2

While new markings exist, do the following:

Step 2.1

Select a “new” marking *M*.

Step 2.2

If *M* is identical to a marking on the path from the root to *M*, then tag M “old” and go to another marking.

Step 2.3

If no transitions are enabled at *M*, tag *M* “dead-end”.

Step 2.4

While there exist enabled transitions at *M*, do the following for each enabled transition at *M*.

Step 2.4.1

Obtain the marking M that results from firing *t* at *M*.

Step 2.4.2

On the path from the root to *M* if there exist a marking *M''* such that *M'* (p) ≥ *M''* *(p)* for each place p and *M' ≠M''*, That is *M''* is coverable, then replace *M' (p)* by ω for each p such that *M' (p)* ≥ *M '' (p)*.

Step 2.4.3

Introduce *M'* as a node, draw an arc with label *t* from *M* to *M'*, and tag *M'* "new".

Consider the example in Figure 3.4

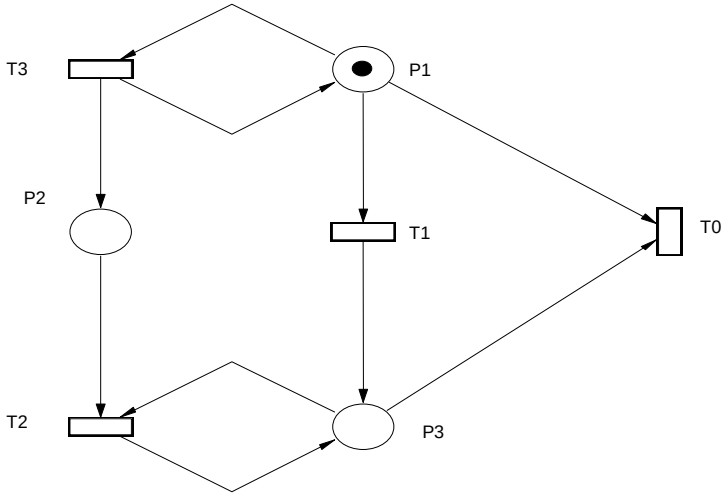

Figure 3.4

The Petri Net shown above (figure 3.4) has transitions T0, T1, T2, T3 and T4.Transitions T0, T1, T2, T3 are dead (L0-live), L1- live, L2-live and L3-live respectively.

The diagrams below describe the Coverability Tree and Coverability Graph of the Net shown in figure 3.4.

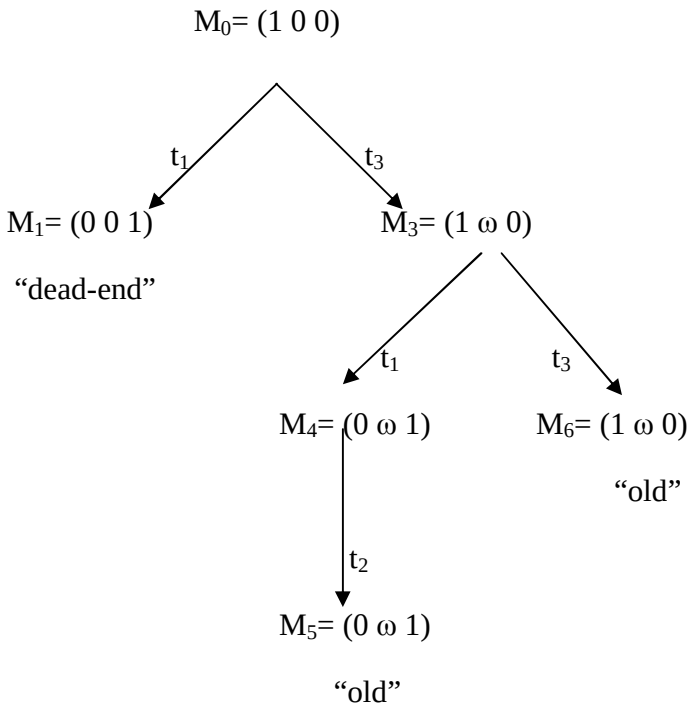

Figure 3.5 Coverability Tree

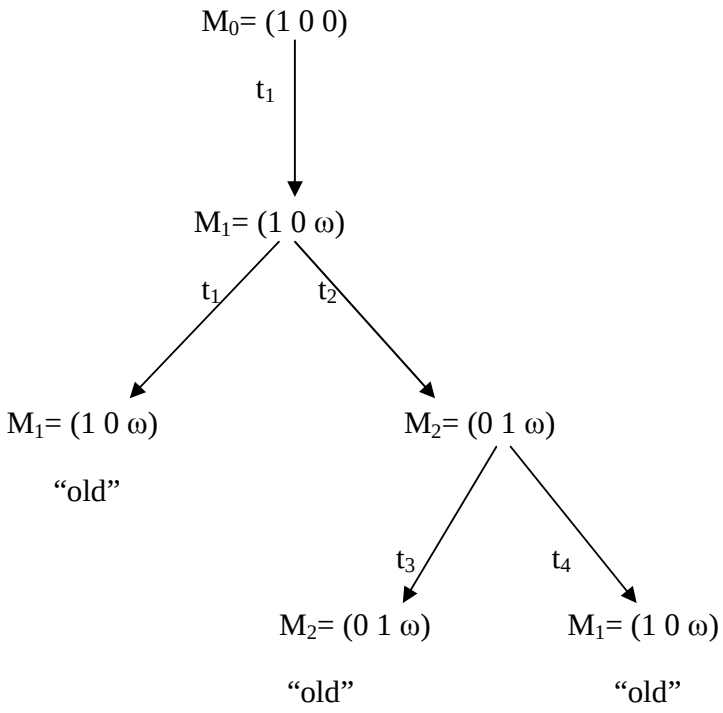

Figure 3.6 Coverability Graph

4. Modeling Using UML

4.1 Modeling the ATM System

A model is a representation of the construction and working of some system of interest. It is similar to but simpler than the system it represents. One purpose of a model is to help analysts predict the effect of changes to a system. It should be a close approximation to the real system and incorporate most of its salient features. On the other hand a model should not be so complex that it is impossible to understand and experiment. A good model is a judicious tradeoff between realism and simplicity.

The Unified Modeling Language (UML) is a graphical modeling language that can be used to specify, construct and document the various parts of a software system. However within the context of this thesis we will use

- The class diagrams to describe the static structure of objects and their relationships
- Use Case to describe interactions of the ATM system with its environment.
- Sequence diagrams to describe the exchange of messages in time within a set of objects.

Modeled Features

The ATM will service a customer at a time. A customer will be required to insert a valid ATM card and enter his Personal Identification Number (PIN).The PIN details will be sent to the bank for validation. Once the customer's PIN is approved he can perform his transaction(s).In the event that the PIN is found to invalid, the customer is given three

attempts to re-enter his password. If all the attempts fail, the card is then ejected out of the ATM.

A selection of activities provided by our ATM includes:

- Cash Withdrawal - Customers should be able to withdraw cash in multiples of two (2) hundred Czech koruna (CZK 200) from any account linked to the ATM card. The amount to be withdrawn by each customer should not exceed the maximum withdrawal limit per day and bank approval is required.
- Balance Inquiry – Customer should be able to check the balance on any account linked to the ATM card.
- Funds Transfer – Enable customer to transfer funds between any two(2) accounts linked to the ATM card
- “Other” (Mobile credit Top Up) – Perform other miscellaneous transactions such as mobile credit top up.

4.2 ATM Class Diagram

The diagram on page 35 (Figure 4.1) depicts identified objects and their attributes and methods or operations associated with them are shown in table 4.1

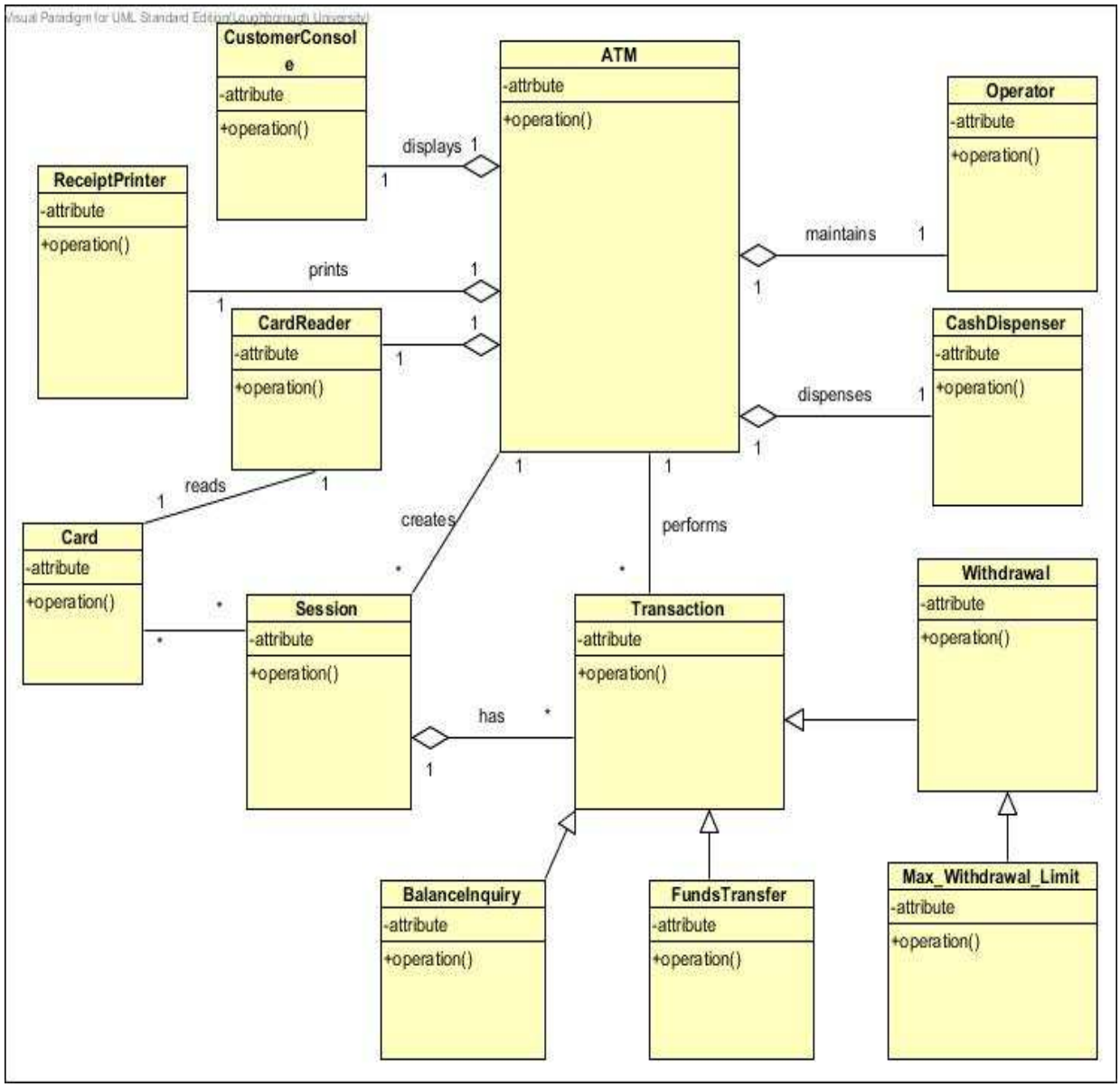

Figure 4.1 ATM System

Name of Class	Attribute	Method(s)
ATM	atmid: integer, bankname: string, atmlocation: string, province: string	createSession(), getLogDetails(), startup(), shutdown(),
CustomerConsole	atm:ATM	displayMenu(),displayMessage(), readPin()
CardReader	atm:ATM	readCard(),ejectCard()
CashDispenser	initialCash:integer totalCash:integer	setInitialcash(), chechMaxcash(), dispenseCash()
Operator	atm:ATM	switchOn(), switchOff(), checkAtmStatus()
Session	atm: ATM, pin:integer, state: string	createSession(), verifyPin()
Transaction	atm:ATM, session: Session, pin:integer ,balance :integer	createTransaction()
FundsTransfer	amount: integer, bankname : string pin:integer	getDetails(),performTransfer()
Withdrawal	amount: integer, bankname : string, pin:integer, balance integer	getDetails(),performWithdrawal()
BalanceInquiry	pin:integer bankname: string	getDetails(),performEnquiry()
Max_Withdrawal_Limit	withdrawal: Withdrawal	chechMaxLimit()

Table 4.1 ATM Classes, Attributes and Methods

4.3 Use Case ATM

The Use Case diagram in Figure 4.2 below will enable a user to transfer funds between accounts linked to his card, perform balance enquiry, withdraw cash from his account and also perform mobile credit top up.

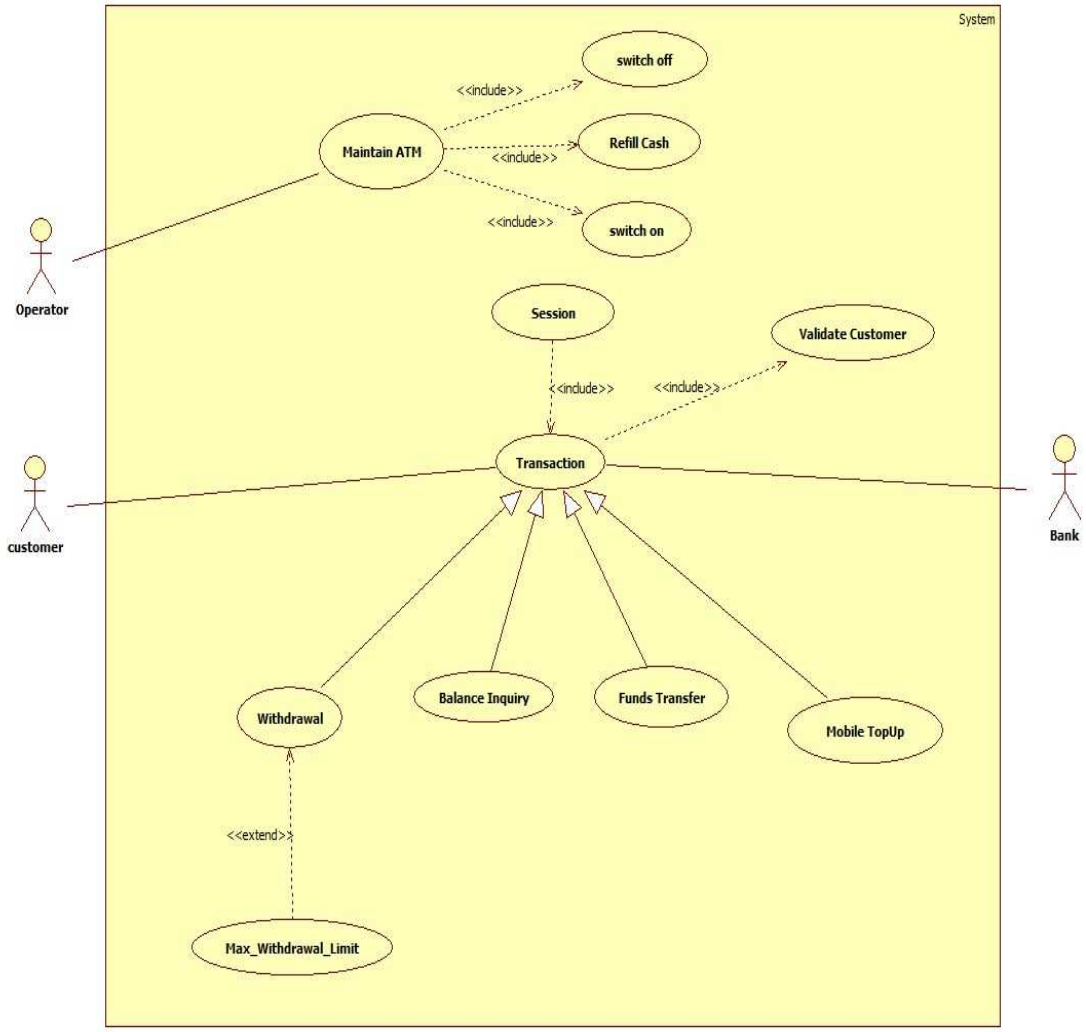

Figure 4.2 Use case for ATM System

The tables below represents the scenarios that takes place in the withdrawal, funds transfer, balance inquiry and mobile top-up Use Case transactions.

Withdrawal Transaction Use Case	
Pre-Condition	
	Customer's ATM card and PIN should be valid.
	There is an active network connection to the Bank.
	ATM has available cash.
Post-Condition	
	Customer receives the cash amount he wants to withdraw and a receipt for his transaction.
	The bank server updates the customer's account balance.
Specification	
Primary Actor	Customer
Stakeholders	
Customer	Requires quick and accurate cash withdrawal service.
Bank	Provides fast, accurate and reliable service to customers.
ATM Consortium	Ensures that the customer is charged the correct amount of surcharge on the withdrawal transaction.
ATM Operator	Ensures that the ATM has sufficient cash in dispenser to support the number of transactions for the day.
Basic Flow of Events	
	Customer inserts his ATM card in the machine .The system prompts the customer to select a language of his choice after which he enters his PIN
	The system validates his ATM card and PIN.
	Customer selects the cash withdrawal option from the transaction menu
	Customer selects the amount he wants to withdraw from the withdrawal menu. If the amount is not specified, the customer selects the "Other" option and then enters his desired amount.
	The system validates the amount entered, checks customer's account balance and verifies whether the machine has enough cash for the transaction, and asks the customer if he or she wants a receipt for the transaction.
	Customer selects "YES" on the receipt screen.
	The system ejects the ATM card, provides the cash, prints the receipt and updates the account balance of the customer in the system.
	Customer takes his ATM, cash and receipt.
Alternative flow of Events	
	If customer has entered invalid PIN. The system prompts the user to re-enter the PIN again. However the system aborts the transaction if an invalid PIN is entered for three (3) consecutive attempts.
	If ATM card is not compatible. The system rejects the ATM card and displays an error message.

Table 4.2

Funds Transfer Transaction Use Case	
Pre condition	
	Customer has more than one (1) active account linked to the ATM card.
	Customer has sufficient cash in the account from which the transfer will be effected.
	Active network connections to the bank exist.
Post condition	
	Funds are transferred from one account to the other and a receipt printed for the transaction.
	The bank server updates the balance of the customers accounts involved the funds transfer transaction.
Specification	
Primary Actor	Customer
Stakeholders	
Customer	Requires smooth and accurate fund transfer service.
Bank	Provides fast , accurate and reliable service to customers.
ATM Consortium	Ensures that the customer is charged the correct amount of surcharge on the funds transfer transaction.
ATM Operator	Ensures that the ATM has sufficient cash in dispenser to support the number of transactions for the day.
Basic Flow of Events	
	Customer inserts his ATM card in the machine .The system prompts the customer to select a language of his choice after which he enters his PIN.
	The system validates his ATM card and PIN.
	Customer selects the Funds Transfer option from the transaction menu.
	Customer selects the account to transfer money from (source account) and the account to transfer money to (destination account). He then specifies the amount of funds to be transferred between the selected accounts.
	The system verifies if account from which funds will be transferred has sufficient funds to undertake the transfer.
	System ejects ATM card, prints receipt for customer and updates the account balance of the customers accounts involved in the funds transfer transaction.
Alternate Flow of Events	
	If customer has entered invalid PIN. The system prompts the user to re-enter the PIN again. However the system aborts the transaction if an invalid PIN is entered for three (3) consecutive attempts.
	If ATM card is not compatible. The system rejects the ATM card and displays an error message.
	Insufficient funds in source account. The system displays an error message and aborts the transaction.

Table 4.3

Balance Inquiry Transaction Use Case	
Pre-Condition	
	Customer's ATM card and PIN should be valid.
	There is an active network connection to the Bank.
Post-Condition	
	Customer receives his/her account balance.
Specification	
Primary Actor	Customer
Stakeholders	
Customer	Requires quick and accurate balance enquiry on his/her account.
Bank	Provides fast, accurate and reliable service to customers.
ATM Consortium	Ensures that the customer is charged the correct amount of surcharge on the balance Inquiry transaction.
ATM Operator	Ensures that the ATM has sufficient cash in dispenser to support the number of transactions for the day.
Basic Flow of Events	
	Customer inserts his ATM card in the machine .The system prompts the customer to select a language of his choice after which he enters his PIN.
	The system validates his ATM card and PIN.
	Customer selects the balance Inquiry option from the transaction menu.
	ATM displays account balance to customer. (Customer will be prompted to select the account that he wants to inquire his balance from in case the customer has more than one (1) account linked to the ATM card).
	ATM asks the customer if he or she wants a receipt for the transaction.
	Customer selects "YES" on the receipt screen.
	The system ejects the ATM card and prints the receipt .
	Customer takes his ATM and receipt.
Alternative flow of Events	
	If customer has entered invalid PIN. The system prompts the user to re-enter the PIN again. However the system aborts the transaction if an invalid PIN is entered for three (3) consecutive attempts.
	If ATM card is not compatible. The system rejects the ATM card and displays an error message.

Table 4.4

Mobile Top-Up Transaction Use Case	
Pre condition	
	Customer's ATM card and PIN should be valid.
	Customer has sufficient cash in his/her account to be used to purchase the mobile credit.
	Customer has an active mobile SIM card from a valid mobile telecommunication operator (Vodafone, T-mobile etc…).
	There is an active network connection between ATM and the Bank
	There is an active network connection between the bank and the mobile telecommunication company.
Post condition	
	Customer receives the mobile top-up credit on his/her mobile phone.
Specification	
Primary Actor	Customer
Stakeholders	
Customer	Requires quick and accurate mobile top up service.
Bank	Provides fast, accurate and reliable service to customers.
ATM Consortium	Ensures that the customer is charged the correct amount of surcharge on the mobile top up transaction.
ATM Operator	Ensures that the ATM has sufficient cash in dispenser to support the number of transactions for the day.
Mobile Operator	Tops up mobile credit on customer's phone.
Basic flow of events	
	Customer inserts his ATM card in the machine .The system prompts the customer to select a language of his choice after which he enters his PIN.
	The system validates his ATM card and PIN.
	Customer selects the Mobile top up option from the transaction menu.
	Customer selects the operator and the amount you want to purchase.
	ATM prints a receipt (voucher) that contains the recharge amount, name of operator and the recharge code.
	The system ejects the ATM card .Customer takes his ATM card and mobile top up receipt (voucher).
	Customer loads the recharge code on his mobile phone to update his mobile credit.
Alternative flow of events	
	If customer has entered invalid PIN. The system prompts the user to re-enter the PIN again. However the system aborts the transaction if an invalid PIN is entered for three (3) consecutive attempts.
	If ATM card is not compatible. The system rejects the ATM card and displays an error message.
	Insufficient funds in customer's account. The system displays an error message and aborts the transaction.

Table 4.5

4.4 ATM Sequence Diagram

A sequence diagram describing the exchange of messages between various objects in an ATM session.

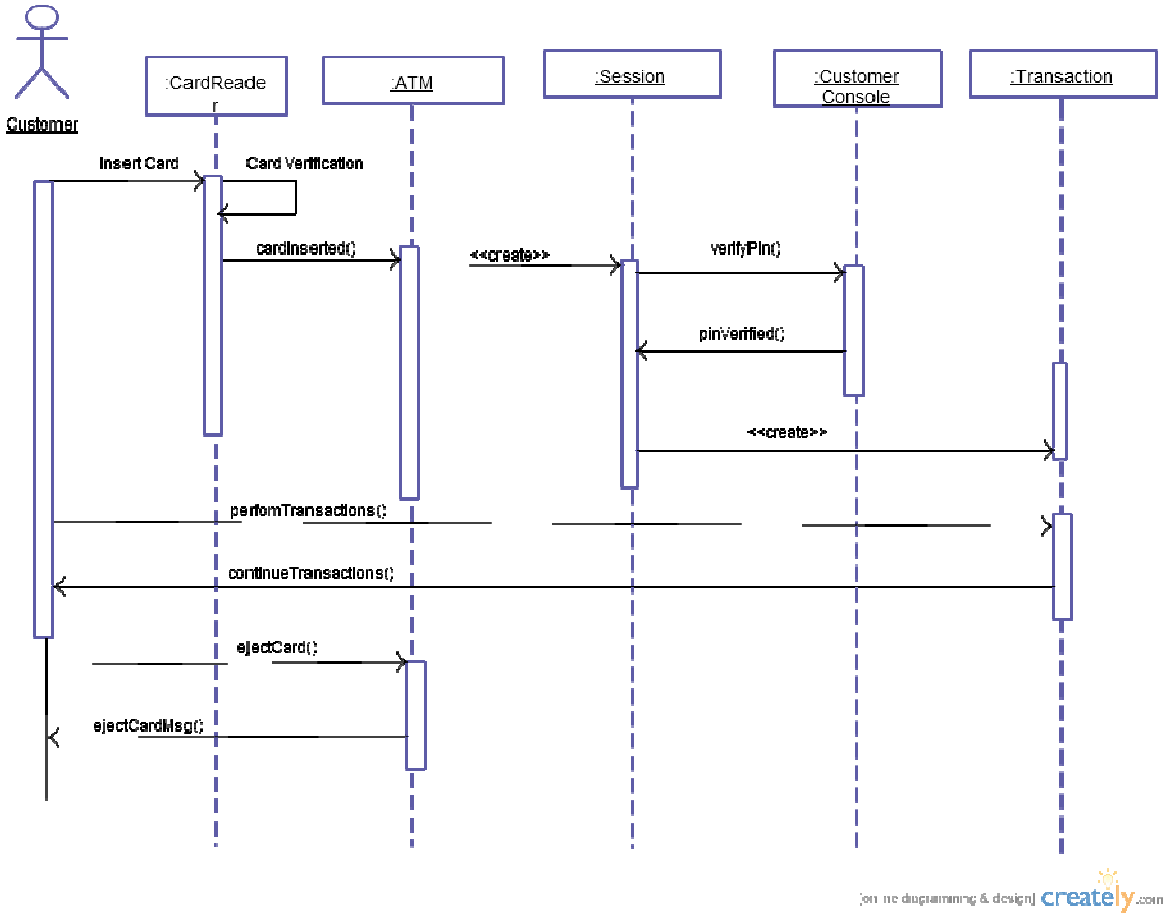

Figure 4.3 Sequence Diagram for ATM Session

5. Modeling and Simulation with Petri Nets

5.1 Petri Net Modeling

The growth in the complexity of modern computer systems such as production, process control, communication systems, etc.., creates numerous problems for their developers. In their planning stages, one is confronted with increased capabilities of these systems due to the unique combination of hardware and software, which operate under a relatively larger number of constraints arising from limited system resources.

Because of the capital intensive and complex nature of these systems, their design and mode of operation requires modeling and analysis to select the optimal design and operational policy. By so doing, flaws detected in the modeling processes are corrected which in the long-run contributes to the operational efficiency of these complex systems.

The purpose of modeling a system is to capture and abstract as much useful information from that system as possible, in such a way as to permit analysis of the system. A Petri Nets ability to model systems that exhibit concurrency, conflict, synchronization, merging etc… have already been discussed in chapter three (3).

As a graphical tool, Petri Nets can be used as a visual-communication aid similar to flow charts, block diagrams and networks. These nets contain tokens that are used in simulating the dynamic and concurrent activities and behavior of systems.

Petri Nets also provide a uniform environment for modeling, formal analysis and design of discrete event systems. A major advantage of using Petri Net models is that the same model is used for the analysis of behavioral properties and performance evaluation.

The Petri Net for the Automated Teller Machine operations is shown in the Figure 5.1, Figure 5.2, Figure 5.3 and Figure 5.4. For each of these Petri Nets, detailed description of the activities have been indicated.

Activities modeled using Petri Net includes the following:

- Withdrawal
  - Enables customer to withdraw cash.
- ATM Funds Transfer
  - Transfer funds between accounts linked to the ATM card.
- Balance Inquiry
  - Provides customer with the available balance on his account(s)
- Mobile Top-Up
  - Purchase credit for his mobile phone.

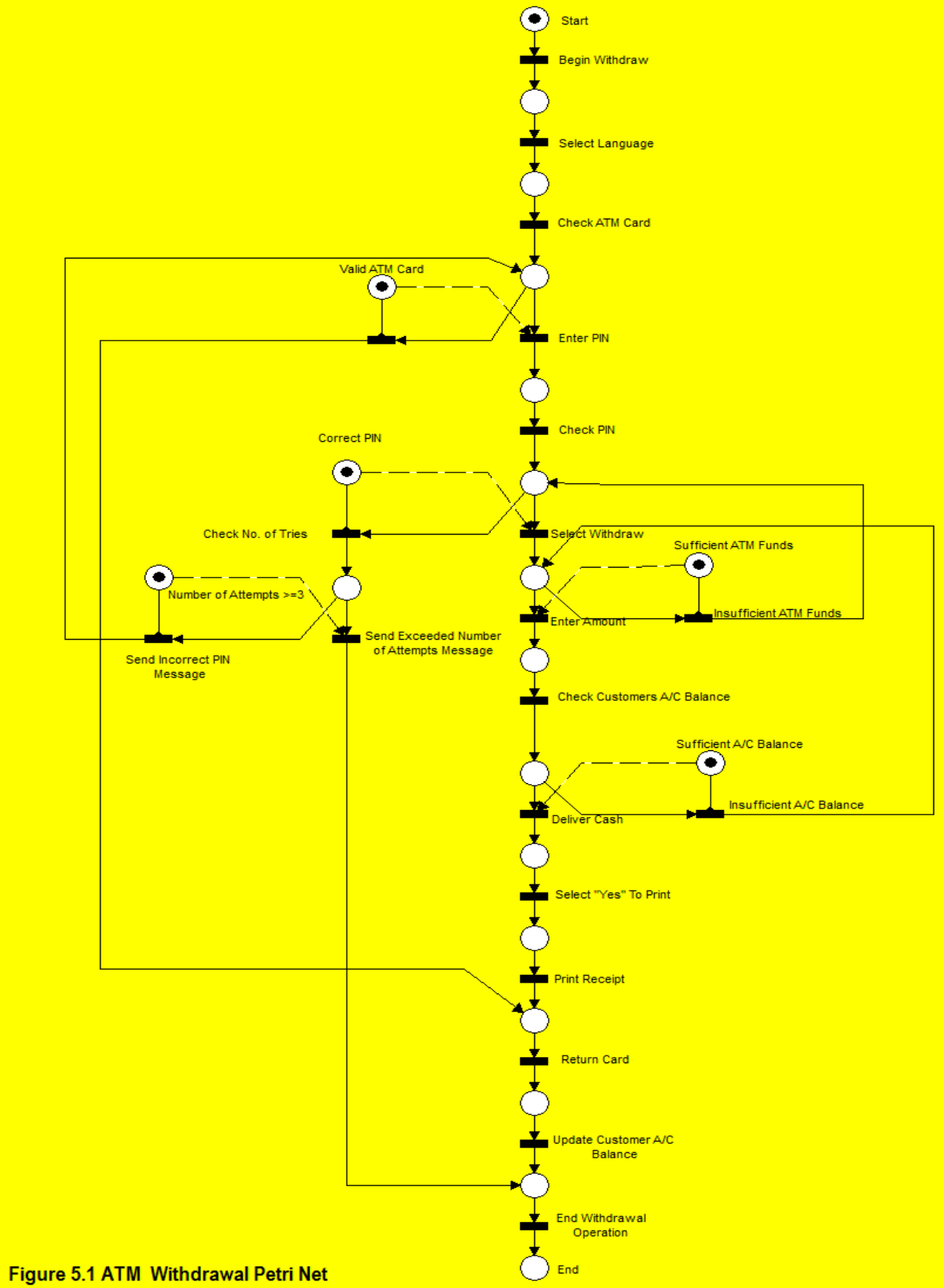

Figure 5.1 ATM Withdrawal Petri Net

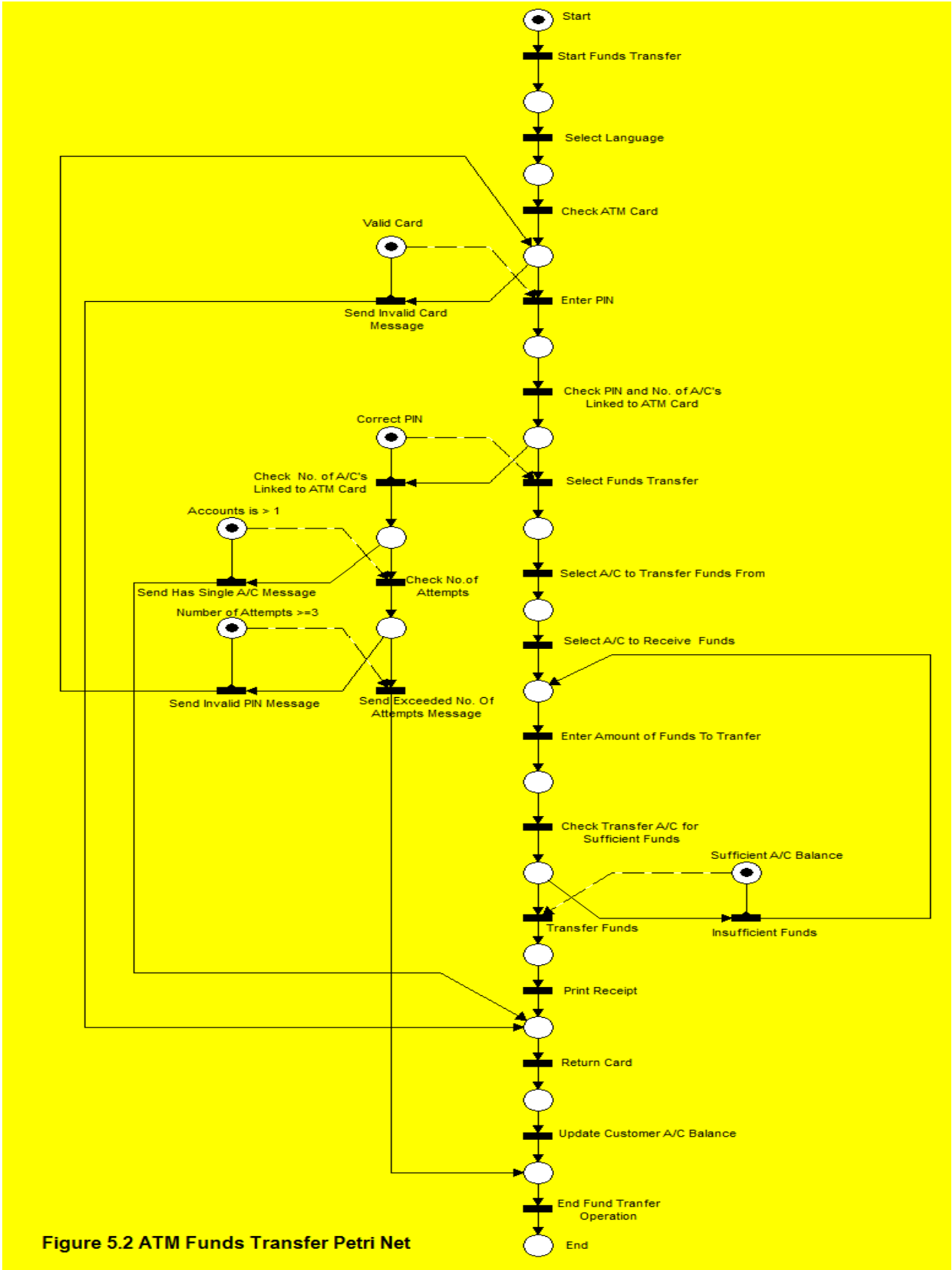

Figure 5.2 ATM Funds Transfer Petri Net

Figure 5.3 ATM Balance Inquiry Peti Net

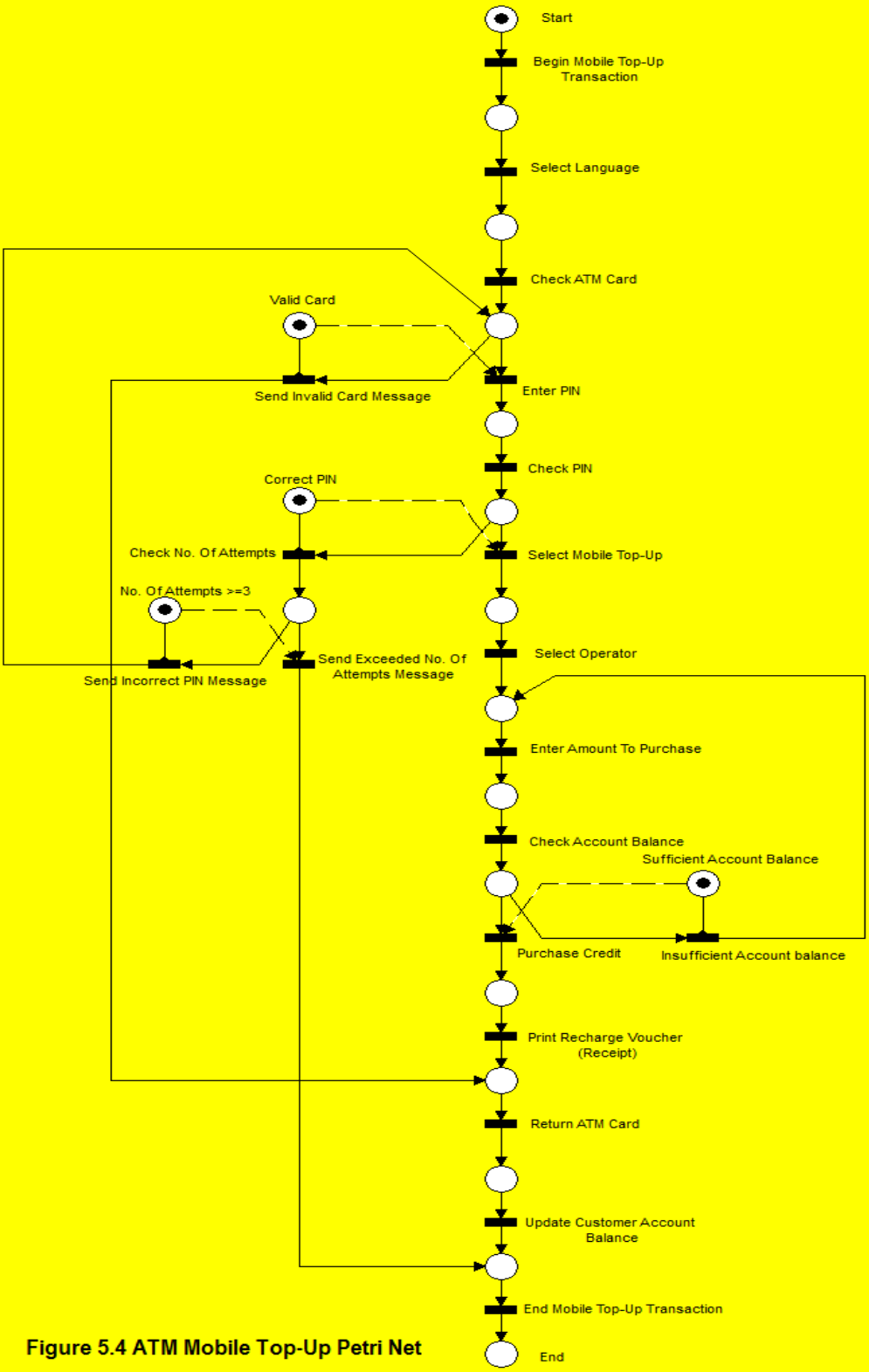

Figure 5.4 ATM Mobile Top-Up Petri Net

5.2 Petri Net Simulation

Simulation is generally defined as a computerized version of a model which is run over time to study the implications of the defined interactions. Simulations are generally iterative in their development. After a model is developed, we may simulates it, learn from the simulation, revise the model if it's required and continue the iterations until an adequate level of understanding is developed.

One of the primary advantages of simulations is that they are able to provide users with practical feedback when designing real world systems. This allows the designer to determine the correctness and efficiency of a design before the system is actually constructed.

Consequently, the user may explore the merits of alternative designs without actually physically building the systems. By investigating the effects of specific design decisions during the design phase rather than the construction phase, the overall cost of building the system diminishes significantly.

Another benefit of simulations is that they permit system designers to study a problem at several different levels of abstraction. By approaching a system at a higher level of abstraction, the designer is better able to understand the behaviours and interactions of all the high level components within the system and is therefore better equipped to counteract the complexity of the overall system.

This complexity may simply overwhelm the designer if the problem had been approached from a lower level. As the designer better understands the operation of the higher level components through the use of the simulator, the lower level components may then be designed and subsequently simulated for verification and performance evaluation. The entire system may be built based upon this top-down technique. This approach is often referred to as hierarchical decomposition and is essential in any design tool and simulator which deals with the construction of complex systems.

Thirdly, simulations can be used as an effective means for teaching or demonstrating concepts to students. This is particularly true of simulators that make intelligent use of computer graphics and animation. Such simulators dynamically show the behaviour and relationship of all the simulated system's components, thereby providing the user with a meaningful understanding of the system's nature.

The ability to simulate using Petri Nets helps us in understanding the behaviour and interaction between the various states and events within the ATM system. The attached CD called ***Petri*** contains the files for simulating each transaction i.e. Withdrawal, Funds Transfer, Mobile Top-Up and Balance Enquiry.

To view or simulate a particular ATM Transaction, please follow the following steps.

1. Open the Folder called Hpsim1_1 on the CDROM.

2. You can also download HPSim at http://www.winpesim.de/3.html

3. Double click on file called HPSim to open the Petri Net software called HPSim.

4. The Withdrawal, Funds Transfer, Mobile Top-Up and Balance Enquiry file have been named ATM_withdrawal.hps, ATM_funds_transfer.hps, ATM_Mobile_Credit_TopUp.hps and ATM_Balance_Inquiry.hps respectively.

5. To view any of the files in step three (3), simply Click on ***File***, Select ***Open***, Choose the folder called Thesis and then select the appropriate file to view its Petri Net.

6. You can select ***Simulation*** on the menu bar to simulate the Petri Net. Clicking on Reset submenu on the Simulation menu can also will reset the tokens in the Petri Net and allow you to repeat the simulation again. The Producer-Consumer Petri Net named Prod_cons.hps is also available in the folder called ***Thesis.***

6 Optimising ATM Resources

6.1 Producer-Consumer Problem

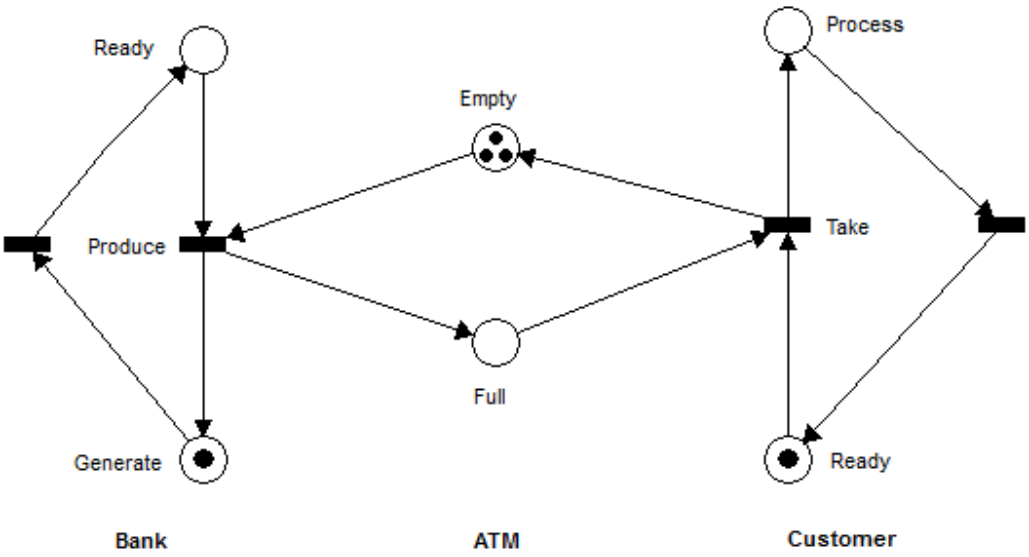

Figure 6.1 Producer-Consumer Petri Net

It is possible that a single place may contain multiple tokens at one time. This situation occurs frequently in dealing with resource allocation problems where there are multiple units of a given resource to allocate. In these problems, a place typically represents the number of available units of the resource.

The specific number of units available at a given time is denoted by the number of tokens contained in the place. When there are no tokens in the place, meaning that there are no available units, activities which need a unit of the resource to execute must wait until a unit is produced. For example, in a distributed system, a sender may generate many data packets that are waiting to be read by the receiver.

The classical producer-consumer problem is a resource allocation problem with a variable number of resources that are limited to a maximum number. There is a producer that generates new units and makes them available to a consumer. The consumer takes one unit of the resource at a time.

The primary synchronization constraints are:

- *Overflow*: the producer cannot produce a new unit unless the number of units is below the maximum number allowed, and
- *Underflow*: the consumer cannot take a unit unless there is at least one available.

The Petri Net shown in Figure 6.1 is a model of a producer-consumer system where the maximum number of units is limited to 3. In this model, there are two places that are used to represent the number of units produced but not yet consumed and the number of additional units that can be produced. These places are named *Full* and *Empty*, respectively. The names *full* and *empty* reflect that the units are often contained in a fixed sized buffer of size N, where N is the maximum number of units allowed.

The number of buffer entries that are full contain produced units that are available to the consumer (customer) and the number of buffer entries that are empty contain spaces available to the producer (bank) to store new units. An invariant in this model is that number *(full) + number (empty) = N*. The buffer is modeled in Figure 6.1 by the two places in the middle named *Empty* and *Full*.

The three tokens in the *Empty* place represent the initial state of the system with three empty buffer elements. The absence of tokens in the *Full* place represents the initial state of the system where there are no buffer elements with information.

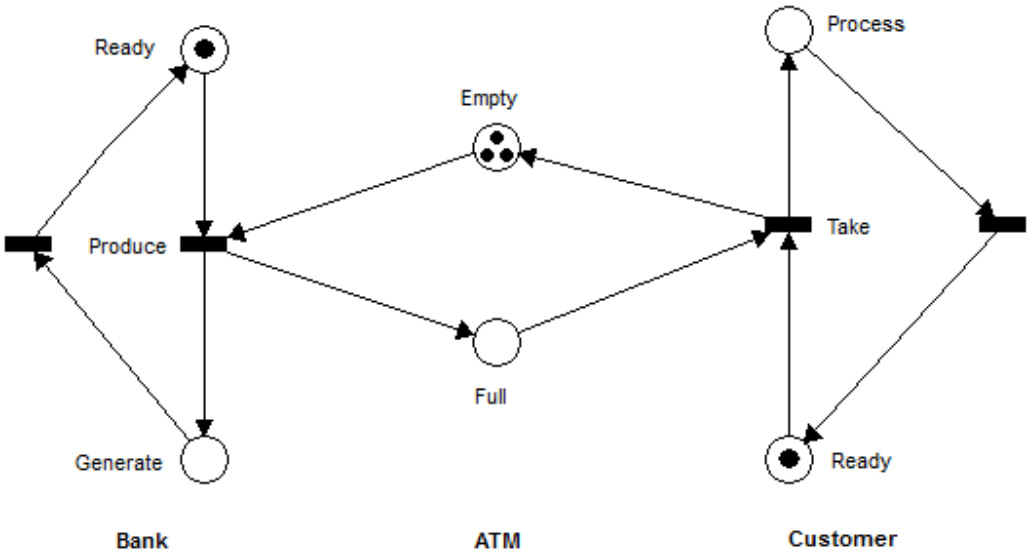

Figure 6.2 Producer (bank) is ready to insert into *Full* place

The producer (bank) in Figure 6.1 is modeled as a subsystem with two places. The *Generate* place represents the condition of the producer when it is generating the next unit of

information to transmit to the consumer. The *Ready* state represents the condition where the producer is ready to insert the newly generated information into the buffer where it is will be available to the consumer.

Notice that the transition *Produce* for the producer can only fire when the producer is *Ready* and there is at least one token in the *Empty* place (denoting a currently empty buffer element into which the new information can be placed).

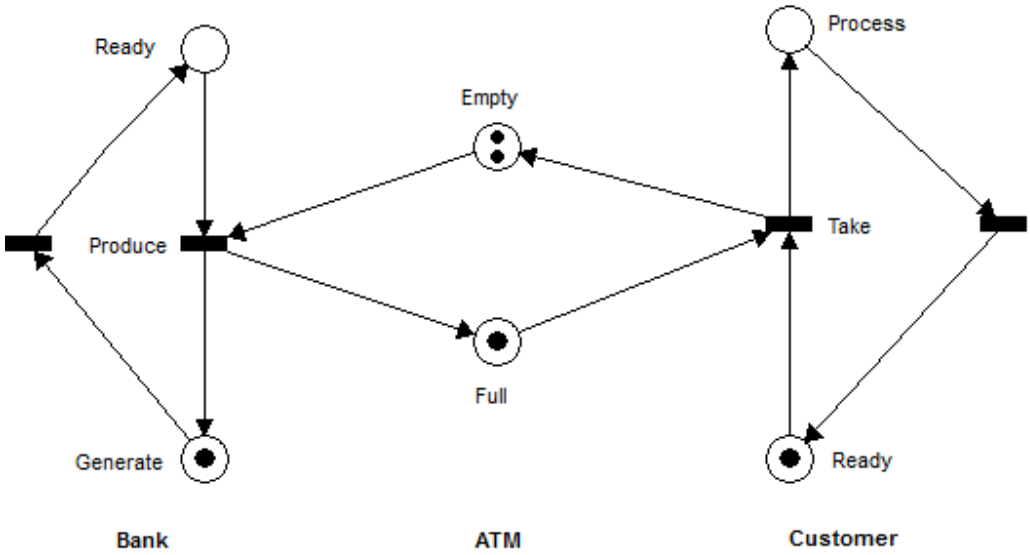

Figure 6.3 One (1) buffer element in *Full* place

The consumer in Figure 6 is also modeled as a subsystem with two places. The *Ready* place represents the condition where the consumer is ready to receive the next unit of new information that was generated by the producer. It is important to note that the transition *Take*

for the producer can only fire when the producer is *Ready* and there is at least one token in the *Full* place (denoting a buffer element containing new information which can be retrieved).

The *Process* place in the consumer represents the condition of the consumer when it is processing the new information most recently retrieved from the buffer.

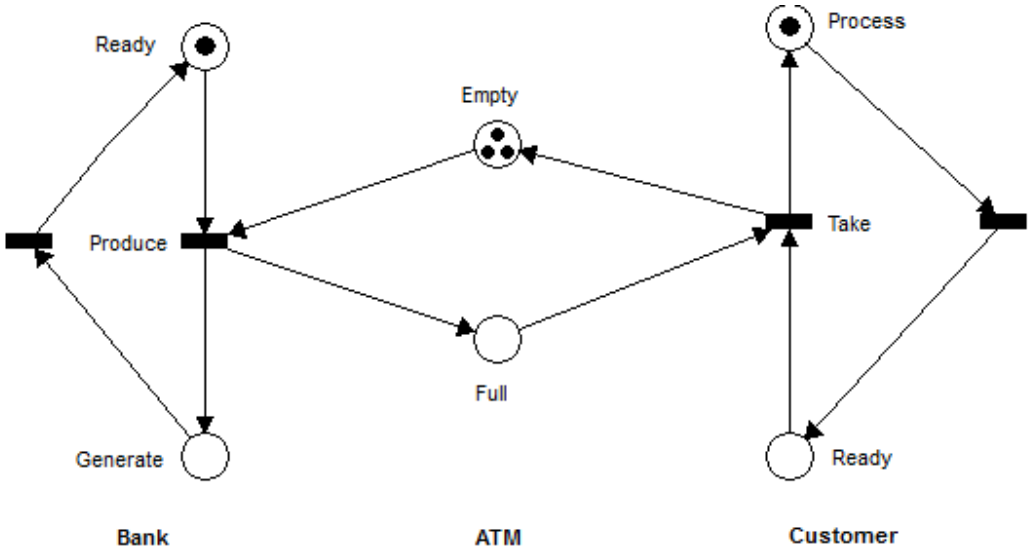

Figure 6.4 Producer prepares to insert new data into buffer whilst the Consumer processes information recently retrieved from buffer

It can be observed that the producer-consumer system in Figure 6 satisfies the two primary synchronization constraints noted above. The overflow constraint is satisfied because the producer cannot fire its *Produce* transition unless there is at least one token in the *Empty* place. Thus, it is not possible for the *Produce* transition to fire four (or more) times in a row without the *Take* transition firing one or more times.

The underflow constraint is satisfied because the consumer cannot fire its *Take* transition unless there is at least one token in the *Full* place. Thus, it is not possible for the *Take* transition to fire four (or more) times in a row without the Produce transition firing one or more times.

6.2 ATM Resources Allocation And Management

A Bank's ATM provides its customers certain services which may include the following:

- cash withdrawal
- Funds Transfer
- Balance Enquiry
- Mobile Top-Up

For a typical cash withdrawal service, a bank keeps a specified amount of cash in the ATM's casket. Since a place represent the available number of the units of resource (tokens) available at a given time, the absence of a token in a place means that there are no available units. Hence activities that need a unit these resources to execute must wait until a unit of these resources are returned or produced.

Clearly, resources required for the execution of the above mentioned ATM activities must be available before the activities can be executed. An example is the cash withdrawal

activity. A customer who wants to withdraw a specified amount of money (amount must be less than daily withdrawal limit) may not be able to do so if the resource is unavailable.

The unavailability of the resource could be as a result of insufficient cash in the ATM's money casket or a break in the communication link between ATM and bank server amongst other factors. As result, a customer may have to wait until the resource required is produced before he can consume such a resource.

By observing the waiting time that a consumer must wait before a given unit of a resource required is produced, a bank can manage its ATM resource effectively by increasing the available number of units of resource (tokens) available at a place as well as the rate of producing the resource in order the match the consumption rate. As a result of this decision, activities that need a unit of these resources to execute will readily have these resources when needed or may wait for a lesser period of time.

Additionally, careful investigation by the management of the bank to find the cause of longer waiting times by consumers can assist the banks to come up with measures aimed at enhancing efficient use of the bank's ATM resources.

Some of the measures may include:

- Reducing the mean time to recover (MTTR) and increasing the mean time between failures (MTBF). A resource with a higher mean time between failure value means the resource is highly reliable.
- Efficient monitoring of ATM usage by checking ATM logs on a regular basis.
- Develop an effective ATM maintenance (preventive) schedule.

6.3 Conclusion

This primary focus of this thesis was to model and simulate selected activities of the Automated Teller Machine. Using the Unified Modeling Language and Petri Nets, we designed a set of models for the ATM system. The Automated Teller Machines (ATM) activities modeled namely; withdrawal, funds transfer, mobile top-up (credit) and balance enquiry have also been shown.

UML as a standard modeling notation combines techniques from data modeling, business modeling, object modeling and component modeling although it lacks the ability to simulate these models most especially in the area of quantified approach. Petri Nets on the other hand has simulation capabilities in addition to their ability to model discrete event systems.

For each of these Nets, a detailed description of the activities has been indicated. Places representing states or conditions and transitions indicating events have been shown. The places in these nets contain tokens that are used in simulating the dynamic and concurrent activities and behavior of systems. The tokens represent resources held in these places.

Again, each of these ATM operations can also be said to be a business process since they have two unique elements called start and end with a set of events connecting them. By visual inspection of the individual Petri Nets, basic programming constructs such as loops and conditional tests are visible.

Simulating these activities using the HPSim software provide a step by step account of the sequence of events as well as changes in their states. Additionally, the Producer-Consumer Problem demonstrated how the ATM resources could be allocated and managed efficiently such that customers achieve satisfaction with the ATM service provided by their bank.

Further studies could be carried out focusing on the behavioral properties and analysis methods in these discrete event-driven systems whose examples include industrial automated systems, communication systems and computer based systems.

7. References

[1] T. Murata: “Petri Nets: Properties, Analysis and Applications”, *Proceedings of the IEEE,* Vol.77, No.4, 1989, pp. 541-580

[2] Anschuetz, H. “A tool for design and simulation of Petri Nets: HPSIM version 1.1”, s.1.:s.ed. 2001, Available at < http://www.winpesim.de/3.htm>

[3] Claude Girault, Rudiger Valk, “Petri Nets for Systems Engineering”, Springer, 2002

[4] Reisig, Wolfgang, “Petri Nets, An Introduction”, Springer Verlag, 1981

[5] Vedran Kordic, “Petri Nets: Theory and Applications”, I-Tech education and Publishing, Vienna, Austria, 2008

[6] J. Vanicek, M.Papik, R. Pergl, T.Vanicek, “Mathematical Foundations of Computer Science”, Prague 2008

[7] Richard Zurawski, MengChu Zhou, “Petri Nets and Industrial Applications: A Tutorial”. s.l.: IEEE Transactions on Industrial Electronics, 1994, Vol. 41.

[8] Jose Reinaldo. Silva, Eston Almqnca dos Santos, “Applying Petri Nets to Requirements Validation”, ABCM Symposium Series in Mechatronics – Vol. 1 pp. 508-517, 2004

[9] James Lyle Peterson,: “Petri Net Theory and the Modeling of Systems” Prentice Hall - 1981

[10] Prof. Ing. Vrana DrSc, “Projecting of Information Systems with UML”

[11] Anu Maria, “Introduction to Modeling and Simulation”, State University of New York at Binghamton, USA

Available at : <http://www.inf.utfsm.cl/~hallende/download/Simul-2-2002/Introduction_to_Modeling_and_Simulation.pdf

[12] http://www.banknetindia.com/atm/atm.htm

[13] http://ezinearticles.com/?A-Brief-Introduction-to-the-Automated-Teller-Machine&id=5397483

[15] http://www.gii.in/India/history-of-atm/

[16] http://www.blinman.com/atm.jpg

[17] http://www.atm24.com/

[18] http://www.techfak.uni-bielefeld.de/~mchen/BioPNML/Intro/pnfaq.html

[19] www.creately.com

[20] Dennis Kafura, "Notes on Petri Nets", pp 7-9 Available at:

< http://happy.cs.vt.edu/highschool/F2011/CT/Petri-Net-Notes-Expanded.pdf>

[21] European Power Suppliers Manufacturers Association, "Guidelines to Understanding Reliability Prediction", page 4, 24 June, 2005 Available at: <htttp://www.epsma.org/pdf/MTBF%20Report_24%20June%202005.pdf>

[22] Pecht, M.G., Nash F.R., "Predicting the Reliability of Electronic Equipment", Proceedings of the IEEE, Vol. 82, No. 7, July 1994

[23] MIL-HDBK-338B, "Electronic Reliability Design Handbook", October 1, 1998

[24] Vedran Kordic, "Petri Nets: Theory and Applications", I-Tech education and Publishing, Vienna, Austria, 2008

[25] http://www.scribd.com/doc/48046285/Modeling-with-Petri-Nets

[26] Kotonya G, Somerville I., "Requirements Engineering", John Wiley & Sons, 1998

[27] Andrea Bobbio, "System Modeling with Petri Nets", Istituto Elettrotecnico Nazionale Galileo Ferraris Strada delle Cacce 91, 10135 Torino, Italy

[28] Zeigler, Bernard P., “Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems”, Boston, Massachusetts, Academic Press, 1990.

[30] http://www.systems-thinking.org/modsim/modsim.htm

[31] http://web.cs.mun.ca/~donald/msc/node6.html

[32] Rajni Pamnani, Pramila Chawan, Satish Salunke, “Object Oriented Modelling for ATM Systems”, Department of Computer Technology, VJTI University, Mumbai , Available at : <http://www.rimtengg.com/iscet/proceedings/pdfs/se/74.pdf>

[33] http://www.arraydev.com/commerce/jibc/0001-07.htm

[34] http://www.downloadsource.net/2868/Visual-Paradigm-for-UML-/

8. CD-ROM